UML Profile and Metamodel for Services RFP
UPMS
“Services Metamodel”

Overview and Status
Jim Amsden, IBM 28-Sep-2006
**Problem Statement – Standards driven by business needs**

- Globalization, rapid change, the Internet and the “Flat World” stimulating business innovation and integration
  - Each business unit to focus on their key value while leveraging capabilities provided by others for non-core functions
  - Requires business agility to respond to market opportunities and challenges
- Evolution in the business parallels evolution in IT solutions
  - Complexity Management
  - The ability to respond to dynamic change
  - Modularity
  - Encapsulation
  - Separation and integration of concerns
  - Deferred commitment
  - Solutions through composition of other solutions
  - Adaptability
  - Reuse
- Each evolutionary step introduces additional capabilities for separation of concerns, loose coupling, and late binding
Facing Business and IT challenges requires SOA

- A framework for matching needs and capabilities, and combining capabilities to meet needs through services
- A foundation enabling business agility and adaptability
- Promote reuse, growth and interoperability to realize the value inherent in individual assets
- Reflects continuing evolution of computing models to enable reuse and reduce coupling
- Greater focus on separation of concerns and delegation
- Minimize trust and connection assumptions
  - Ownership, distribution and implementation
BPM and SOA can help realize business agility

- Clear separation of the Business operations from services solutions
- Enable integration at the business and IT solution levels independently
- BPM captures and validates business organizational and operational requirements and constraints
- SOA enables flexible solutions to business requirements
But there is a proliferation of protocols, specifications, metamodels and tools

- Many are incompatible with each other
- SOA platforms are in immature and rapidly changing
- OASIS SOA Reference Model has only just been completed
- As a result it will be harder for companies to realize the BPM and SOA potential
- Without costly conversion tools and runtime adapters and mediators
  - Higher development costs
  - Requirement for more skilled developers
  - Bloat in both development and runtime platforms
  - Increased potential for bugs
What is needed is a new Services Standard

- Enable interoperability and integration at the model level
- At a higher-level of abstraction separate from platform variability
- Address business integration and service interaction concerns at the architectural level
  - Architecture is the bridge between the business requirements and IT solutions
- Enable SOA on existing platforms through MDA
- Allows for flexible platform choices
- While preventing existing solutions from inhibiting platform evolution
- Leverage and integrate with existing OMG standards for end-to-end lifecycle development and management
Why WS-* is too much and not enough

- Semantically thin specifications
- Rapidly evolving specifications, likely to be more churn
- Relatively low-level abstraction rooted in XML
  - Communication may be local or remote
- Focused on wire protocols, data interchange, and execution environments
- Represent only one of may possible SOA realizations
- Many standards focused on individual technology segments
- Interdependencies and relationships result from overlaps and gaps
  - Leads to vendor variability and interoperability issues
Why UML2 is too much and not enough

- UML2 covers potentially unrelated modeling domains
  - UseCases for requirements
  - Many different styles of behavioral modeling
  - Deployment modeling
  - Many other modeling constructs
  - Not particularly service centered

- Need a realization of SOA reference model independent of SOA implementation strategies

- Need better definition of service contracts independent of SOA design

- Need more formal separation of specification and realization

- Need location and binding information for modeling service interactions (because of reduced coupling in a distributed environment)

- Idempotent, long-running, compensation semantics
Component Based Development

- Separates component specification from realization
  - Clients only depend on the specification (interfaces)
  - Can substitute evolving realizations to fix bugs or add new features
  - Specification captures one set of concerns
  - Realization addresses those concerns while handling others

- Adds ports for better encapsulation and isolation
  - Better decoupling between requestors and providers
  - Component client only depends on what they need not the whole component

- Provides a better unit of reuse
  - Component is an autonomous entity
  - Specifies what it provided and what is necessary for its use
  - More formal support for commonality and variability
Service Oriented Architectures were introduced to:

- Addresses the effect of application integration across ownership boundaries
- Use Service Level Agreements to capture contracts
- Extends CBD with additional distributed computing and deployment concerns
- Provide more reflective and dynamic systems
  - Behavior can come and go
  - Clients query for service with acceptable QoS
  - Raise exceptions if none found
- Include concepts for publishing, finding, and dynamically binding to services
- Driven by practical implications of the Web and existing middleware platforms
  - Integration between J2EE and .Net
This is a good time for OMG to enter the SOA landscape

- There is common recognition of Business value of SOA
- The importance of WS-* or Web Services as enabling SOA technology is well established
- There are any existing OMG standards that are applicable to SOA
- There is an opportunity for OMG to contribute in order to:
  - Make it SOA easier to development, understand and manage
  - Provide a more stable SOA environment (through abstraction and separation of concerns)
  - Enable business value through standards for agile processes and supporting technologies
- This will be necessary to achieve:
  - Interoperability necessary for business integration
  - Growth in marketplace of reusable services
Goals of the RFP

- A common vocabulary and metamodel to unify the diverse service definitions that exist in the industry.
- Clarify UML semantics concerned with services modeling and establish modeling best practices.
- Complement existing UML metamodel by defining an extension to UML to ensure complete and consistent service specifications and implementations.
- Integrate with and complement standards developed by other organizations such as W3C and OASIS.
- Support a service contract describing the collaboration between participating service consumers and service providers
  - clearly separate service requirements and specification from realization.
- Enable traceability between contracts specifying services requirements, service specifications that fulfill those requirements and service providers that realize service specifications.
- Facilitate the adoption of Service Oriented Architectures through
  - more abstract and platform independent services models to speed service development,
  - decouple service design from evolving implementation, deployment and runtime technologies, and
  - enable generation of platform specific artifacts.
- The ability to exchange services models between tools using XMI.
Out of scope – for future RFPs

**Focus first on Service Capture**
- Methodologies for service design.
- Services governance or compliance.
- Service metrics, policy, security, trust, performance, or other Qualities of Services
- Wire protocols and/or message transfer encodings or marshalling.
- Message delivery reliability, transaction scopes, or other mechanisms for managing data integrity.
- Service brokering, publishing, discovery, service addressing, service registries, asset management.
- Service runtime configuration and deployment.
- Dynamic binding, service federation, mediation, service bus structure, or other service execution concerns.
- User experience or user interfaces.
Where the Services Metamodel fits into SOA
Relationship to existing OMG specifications

- ODM
- BMM, OSM, SBVR, BPMN and BPDM
- UML2, OCL, EDOC
- KDM, IMM
- MOF, QVT, XMI
- ODM, RAS
Relationship to other specifications

- OASIS Reference Model for Service Oriented Architecture
- XSD Specification
- Service Data Objects Specification
- WSDL Specification
- Service Component Architecture Specification
- WSBPEL Specification
- FEA Service Component Reference model (SRM)
- FEA Services and Components Based Architectures (SCBA)
- ebXML
Mandatory Requirements

- MOF metamodel and equivalent UML2 profile
- Extend, but not conflict with UML semantics
- Notation icons for services extensions
- Platform independent
- Non-normative mapping to Web Services
Service Contracts

- Specify service contracts (architecturally neutral)
  - Interactions between service consumers and providers
  - Realize use cases (for requirements)
  - Specified functions
  - Participants and the roles they play
  - Responsibilities of participating roles
  - Behavioral rules for how roles must interact
  - Constraints for objectives that must be met

- Service contract semantically equivalent to BPDM choreography and collaborations

- Service specifications and providers fulfill service contracts

- Loose and strict contract fulfillment

- Use of service contracts is optional
Service Specification

- Separate for how services are provided or implemented
  - Provided and required service interfaces
  - Service operations (distributed, concurrent)
  - Operation pre and post conditions, parameters and exceptions
  - Constraints service providers must honor
  - Interaction points through which consumers and providers connect
  - Behaviors service operation methods indicating required semantics of realizing service providers

- Use of service specifications is optional
Service Data

- Structural information exchanged between service consumers and service providers
- Attachments for opaque information
- Usage semantics make no assumptions with regard to global synchronization, control or shared address spaces
Service Invocation and Event Handling

- Support synchronous and asynchronous service invocation
- Synchronicity is a property of the invocation, not the service definition
  - Clients determine how services are used
- Designate the ability to receive an event
- Generate events targeted at a specific service provider or broadcast to interested providers
- Service operations responding to events are asynchronous, have no outputs, and may raise exceptions
Service Parameters, Consumers and Providers

- Parameter types are primitive types or service data
- Designate service consumer and services required
- Designate service provider and services provided
- Services only provided through interaction points, not direct connections between service consumers and service providers
- Service provider may realize zero or more service specifications
- Service provider must be conformant to all realized service specifications
- Interaction point of a service provider provides and/or requires one or more service interfaces
- Service provider specifies binding information applicable to all interaction points
- Interaction point can restrict or extend service bindings
Service Realizations and Composition

- Specify realizations of provided service operations through owned method behaviors of service provider
- Multiple styles for specifying method behaviors – Activity, Interaction, StateMachine, ProtocolStateMachine, OpaqueBehavior, etc.
- Method behavior style may differ from that used by its specification
- Specify how services are composed from other services
- No assumptions about or constraints on the number of recursive levels of composed services, or arbitrary distinctions between composition levels.
Connecting Service Consumers and Providers

- Service channels for connecting between usages of service consumers and service providers in some containing element

- Support different degrees of coupling between consumers and providers through service provider specified as:
  - A service interface
  - A service specification
  - A particular (concrete) service provider

- Service channel selects from bindings expected by service consumer and provided by service provider
Extensibility and Service Partitions

- Enable customization and extending services through
  - Configuration properties (profile markings)
  - Refinement and redefinition through generalization
  - Pattern or template specification and instantiation
- Put service specifications and/or providers into logical groupings for organization and management
- Specify constraints on service connections between service partitions.
Service Model Interchange

- Service model interchange through XMI
- Service models captured by the services metamodel are exchanged according to MOF-to-XMI mapping rules
- Service models captured by the services UML2 profile are exchanged according to the UML rules for exchanging instances of UML models with applied profiles
- Define interchange compliance levels for each of these XMI document formats
Optional Requirements

- Additional non-normative mappings to existing platforms and languages for service specification and/or execution
- Specify preferred encoding for service data exchange
- Binding metamodel
Issues to discuss

- Relationship of submission and UML to demonstrate semantic consistency
- How the specification supports automated consistency checks for model validation
  - Especially between service contracts and the service specifications and providers that realize them
- Applicability to ESB and common runtime architectures
- Relationship to UDDI
Example – a Purchase Order Process

- Taken from the WSBPEL (BPEL4WS) specification
- Illustrative only – not intended to suggest any particular submission requirements or recommendations
- The intention is only to illustrate and clarify the submission requirements
- Captured using IBM Rational Software Architect
A simple Business Driven Development Process

What are we trying to accomplish and why? How can we know if our objectives have been met?

What are the business organizational and operational requirements necessary to realize these objectives?

How do we ensure we are addressing the right requirements within operational constraints?

How should we realize those operational requirements while addressing IT and business integration concerns?

How do we create solutions based on the chosen architectures?

How can we deliver solutions that respond to rapidly changing business opportunities?

How do we collect data and monitor the results for solution validation and continuous improvement?

What are you trying to accomplish, not how. Includes metrics and KPIs to measure achievement.

Capture business requirements that realize goals and objectives from the perspective of key stakeholders in business use cases.

Model as-is and to-be business processes that realize the business requirements. Use simulation to measure KPIs and verify business goals are met.

Design a system architecture that realizes the business processes while addressing IT and QoS concerns captured in system use cases. View the business processes as business services models. Capture system use cases to specify implementation requirements.

Use MDA to generate the initial IT solution based on the chosen architecture and realizing the system use case requirements.

Integrate new, reused, and purchased services into a completed solution.

Design and implement a deployment strategy that meets nonfunctional requirements for performance, security, availability, maintainability, etc.
Business Motivation/Objectives

A consortium of companies has decided to collaborate to produce a reusable service for processing purchase orders. The goals of this project are to:

- Establish a common means of processing purchase orders.
- Ensure orders are processed in a timely manner, and deliver the required goods.
- Help minimize stock on hand.
- Minimize production and shipping costs
Business Organizational and Operational Requirements

Scheduling

- Request Product Scheduling
- Send Shipping Schedule

Shipping

- Request Shipping
- Process Schedule

Invoicing

- Initiate Price Calculations
- Handle Shipping Price
- Process Invoice
Business Organizational and Operational Requirements
Service Requirements Contract – from the BPMN process

```
<<serviceInterface>>
Shipping
  + requestShipping()

<<serviceInterface>>
Invoicing
  + initiatePriceCalculation()
  + completePriceCalculation()

<<serviceInterface>>
Purchasing
  + processPurchaseOrder()

<<serviceInterface>>
Scheduling
  + requestProductionScheduling()
  + sendShippingSchedule()

<<serviceContract>>
Process Purchase Order

  shipping : Shipping
  invoicing : Invoicing
  orderProcessor : Purchasing
  scheduling : Scheduling
```
The rules for how the role interact
We now switch to modeling the services solution
Production scheduling services

```
<<serviceInterface>>
Scheduling
+ requestProductionScheduling (customerInfo : Customer, purchaseOrder : PurchaseOrder)
+ sendShippingSchedule (schedule : Schedule)
```

```
<<serviceProvider>>
Productions
<<service>> scheduling : Scheduling
```
org::crm::domain defines the domain information model

- Domain data used in the implementation of services
- Entities are often persisted in some data source
- Each entity must have properties that can be used to distinguish different instances
- Some entities can also be used as messages
This domain data can also be used to populate messages

• Messages are data exchanged between service consumers and providers
• Messages may be views on the domain data (selections and projections)
• These views have to be mapped to the domain data somehow
• Service implementations can be responsible for moving data between messages and domain entities
Invoicing services

```
credit <import> crm

<serviceInterface>
Invoicing
+ initiatePriceCalculation ( customerInfo : Customer, purchaseOrder : PurchaseOrder )
+ completePriceCalculation ( shippingInfo : Manifest )

InvoicingProtocol
invoicing : Invoicing
orderer : InvoiceProcessing

InvoiceProcessing
+ <trigger> processInvoice ( invoice : Invoice )

<serviceProvider>
Invoicer

<use>
```
The rules or protocol for orderer interaction with invoicing

- The protocol is captured in an ownedBehavior of the Collaboration that is the type of the port
- It models the conversation, protocol or interaction between consumer and provider
- The protocol could be an:
  - Interaction
  - Activity
  - StateMachine
  - ProtocolStateMachine
  - OpaqueBehavior
- There could be more than one interaction between the consumer and provider
  - The Collaboration would have a different ownedBehavior for each one
- The behavior’s specification is the provided operation that is invoked by the consumer
- The contract for a Connector between the requestor and provider is a CollaborationUse whose type is this Collaboration
Shipping services

```
Shipping
+ requestShipping (customerInfo : Customer, shippingInfo : Manifest)
```

```
Shipper
+ shipping : ShippingProtocol
```

```
ScheduleProcessing
+ processSchedule (schedule : Schedule)
```

```
ShipperImpl
```
The protocol for the shipping service

```
Interaction1

consumer:ScheduleProcessing            shipping:Shipping

1: requestShipping

2: processSchedule
```
OrderProcessor external or “black box” view
The OrderProcessor internal structure or “white box” view
The processPurchaseOrder Service Implementation
Fulfilling the Service Contract

OrderProcessor

contract: Process Purchase Order

- invoicing
  - invoicing

- scheduling
  - scheduling

- shipping
  - shipping

- id: String
  - id

- shippingInfo: Manifest
  - shippingInfo

- schedule: Schedule
  - schedule
Assembling the parts into a Deployable Subsystem
A closer look at invoicing and how the protocol is satisfied
Summary
## Timeline

<table>
<thead>
<tr>
<th>Event or Activity</th>
<th>Actual Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preparation of RFP by TF</td>
<td>Sept 4, 2006</td>
</tr>
<tr>
<td>TC votes to issue RFP</td>
<td>Sept 27, 2006</td>
</tr>
<tr>
<td>LOI</td>
<td>Nov 28, 2006</td>
</tr>
<tr>
<td>Initial submission presentations</td>
<td>June 4, 2007</td>
</tr>
<tr>
<td>Revised submissions due</td>
<td>Nov 19, 2007</td>
</tr>
<tr>
<td>Final Submission</td>
<td>March 2008</td>
</tr>
</tbody>
</table>